**BTVN**

Cho văn bản P: “ toi thay ban than minh o hien tai có nhieu diem khac so voi qua khu ngay ng ova ngo nghech truoc kia toi da khon lon that roi”

Mẫu T:”toi”

Yêu cầu:

1. Cài đặt 2 thuật toán Rabin Karp và KMP xác định vị trí của mẫu T so với văn bản P.
2. Nhận xét về 2 thuật toán

**Rabin Karp**

#include <iostream>

#include <cstring>

#include <time.h>

using namespace std;

// d là số ký tự trong bảng alphabet

#define d 256

void RabinKarp(char *pat*[], char *txt*[], int *q*)

{

    int pat\_len = strlen(*pat*);

    int txt\_len = strlen(*txt*);

    int i, j;

    int p = 0; // giá trị hash của pat

    int t = 0; // giá trị hash của txt

    int h = 1;

    for (i = 0; i < pat\_len - 1; i++)

        h = (h \* d) % *q*;

    // Tính giá trị hash của pat và cửa sổ trượt của txt

    for (i = 0; i < pat\_len; i++)

    {

        p = (d \* p + *pat*[i]) % *q*;

        t = (d \* t + *txt*[i]) % *q*;

    }

    for (i = 0; i <= txt\_len - pat\_len; i++)

    {

        // kiểm tra giá trị hash của cửa sổ trượt của txt và pat.

        // Nếu bằng nhau thì kiểm tra từng ký tự

        if ( p == t )

        {

            for (j = 0; j < pat\_len; j++)

            {

                // có ký tự khác nhau

                if (*txt*[i+j] != *pat*[j])

                    break;

            }

            // if p == t and pat[0...pat\_len-1] = txt[i, i+1, ...i+pat\_len-1]

            if (j == pat\_len)

                cout<<"Pattern found at index "<< i <<endl;

        }

        // Tính giá trị hash cho cửa sổ trượt tiếp theo của txt (trượt qua phải 1 ký tự)

        if ( i < txt\_len - pat\_len )

        {

            t = (d\*(t - *txt*[i]\*h) + *txt*[i+pat\_len]) % *q*;

            // Nếu t < 0, chuyển sang t > 0

            if (t < 0) t = (t + *q*);

        }

    }

}

int main()

{

    char txt[] = "toi thay ban than minh o hien tai có nhieu diem khac so voi qua khu ngay ng ova ngo nghech truoc kia toi da khon lon that roi";

    char pat[] = "toi";

    // q nên là số tự nhiên lớn để giảm đụng độ

    int q = 101;

*clock\_t* start, end;

    start = clock();

    RabinKarp(pat, txt, q);

    end = clock();

    cout << "time: " << (double) (end - start);

    return 0;

}

**KMP**

#include <iostream>

#include <cstring>

#include <time.h>

using namespace std;

void computeLPSArray(char\* *pat*, int *M*, int\* *lps*);

void KMPSearch(char\* *pat*, char\* *txt*)

{

    int pat\_len = strlen(*pat*);

    int txt\_len = strlen(*txt*);

    int \*lps = **new** int[pat\_len];

    // Tính mảng LPS

    computeLPSArray(*pat*, pat\_len, lps);

    int i = 0; // index for txt[]

    int j = 0; // index for pat[]

    while (i < txt\_len) {

        if (*pat*[j] == *txt*[i]) {

            j++;

            i++;

        }

        if (j == pat\_len) {

            printf("Found pattern at index %d\n", i - j);

            j = lps[j - 1];

        }

        // không trùng

        else if (i < txt\_len && *pat*[j] != *txt*[i]) {

            if (j != 0)

                j = lps[j - 1];

            else

                i = i + 1;

        }

    }

}

// Tạo mảng LPS từ pattern

void computeLPSArray(char\* *pat*, int *M*, int\* *lps*)

{

    int len = 0;

*lps*[0] = 0;

    int i = 1;

    while (i < *M*)

    {

        if (*pat*[i] == *pat*[len]) {

            len++;

*lps*[i] = len;

            i++;

        }

        else

        {

            if (len != 0)

                len = *lps*[len - 1];

            else // if (len == 0)

            {

*lps*[i] = 0;

                i++;

            }

        }

    }

}

// Driver program to test above function

int main()

{

*clock\_t* start, end;

    char txt[] = "toi thay ban than minh o hien tai có nhieu diem khac so voi qua khu ngay ng ova ngo nghech truoc kia toi da khon lon that roi";

    char pat[] = "toi";

    start = clock();

    KMPSearch(pat, txt);

    end = clock();

    cout << "time: " << (double) (end - start);

    return 0;

}

1. Nhận xét 2 thuật

|  |  |
| --- | --- |
| **Rabin Karp** | **KMP** |
| Nhanh hơn trong bài tập này | Chậm hơn trong bài tập này |
| Có khả năng xảy ra xung đột trong bảng băm | Ổn định với mọi TH |
| Dễ cài đặt | Khó cài đặt |